

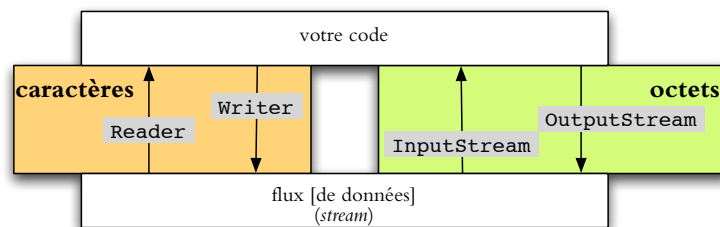
IFT1025 automne 2009

Miklós Csűrös

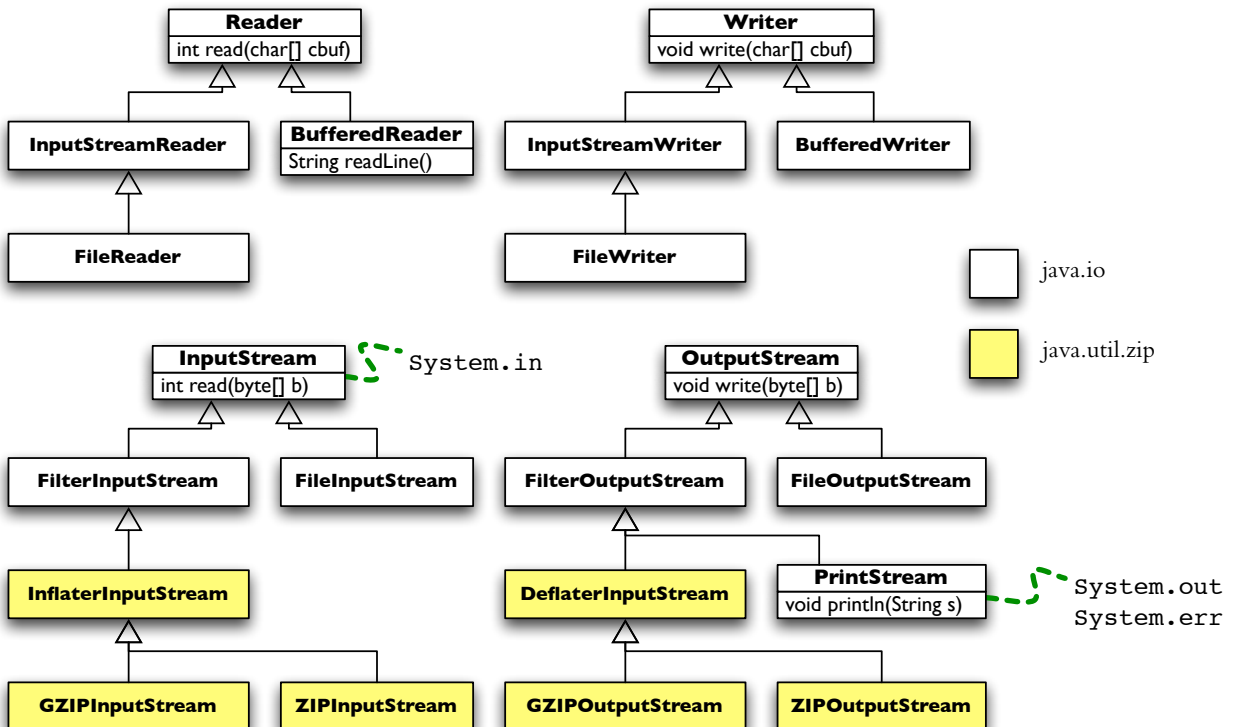
22 septembre 2009

Entrée/sortie

Abstraction entrée/sortie : on travaille avec des **flux** de données. Écriture et lecture en binaire (flux d'octets) ou comme texte (flux de caractères).



Il existe une grande hiérarchie de classes pour entrée/sortie. Quelques exemples :



Suite typique d'opérations sur un fichier :

1. ouvrir ou créer le fichier (par constructeur p.e., `new PrintStream("fichier.txt")`)
2. lecture ou écriture de données (méthodes de la classe p.e., `.println("une ligne")`)
3. fermer (méthode `close()`) → **important!** (relâche les ressources, écriture du tampon)

Les classes de bases (`Reader`, `Writer`, `InputStream`, `OutputStream`) définissent des opérations de «bas niveau» (par caractère ou octet). Les autres classes ajoutent des fonctionnalités telles que l'usage de **tampons** (*buffers*), méthodes de lecture/écriture en formats différents, conversion d'encodage. Pour cela, on utilise du «wrapping» :

```
public void readFile(String file_name)
{
    BufferedReader BR
        = new BufferedReader( // wrapping pour efficacité
            new FileReader(file_name));
    while(true) // condition d'arrêt dans la boucle
    {
        String line = BR.readLine();
        if (line == null) break;
        // parsing
    }
    BR.close();
}
```

fonctionnalité	classe pour wrapping	méthodes
lire ligne par ligne	<code>BufferedReader</code>	<code>readLine()</code>
écrire ligne par ligne	<code>BufferedWriter</code>	<code>write(String s), newLine()</code>
«pretty print»	<code>PrintStream</code> et <code>PrintWriter</code>	<code>println(...)</code> et <code>printf(...)</code>
encodage de types primitifs	<code>DataOutputStream</code> et <code>DataInputStream</code>	<code>writeDouble,...</code>

Accès au système de fichiers : classe `java.io.File`.

- ★ portabilité : constantes (p.e., `File.separator` est `"/"` sur Unix, `"\"` sur PC, `":"` sur Mac), `createTempFile(String p, String s)`
- ★ manipulation du nom et information : `getPath()`, `isDirectory()`, `length()`, ...
- ★ découverte de répertoires : `list()`

Interface graphique : classe `javax.swing.JFileChooser`. Usage (lecture) :

```
JFileChooser fc = new JFileChooser();
int retval = fc.showOpenDialog(parent); // parent est un Component ou null
// retval est un de JFileChooser.CANCEL_OPTION,
// JFileChooser.APPROVE_OPTION,
// JFileChooser.ERROR_OPTION
if (retval == JFileChooser.APPROVE_OPTION)
{
    File file = fc.getSelectedFile();
    ...
}
```

Pour écriture, faire `showSaveDialog(parent)`.